Query-Driven Visualization

Kurt Stockinger, John Wu, John Shalf, and Wes Bethel

Computational Research Division

Lawrence Berkeley National Laboratory

October 2005

VIS 05

MINNEAPOLIS, MN USA

Motivation and Problem Statement

- Too much data.
- Visualization “meat grinders” not especially responsive to needs of scientific research community.
- What scientific users want:
  - Scientific Insight
  - Quantitative results
  - Feature detection, tracking, characterization
  - (lots of bullets here omitted)
- See:

  http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf

  http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf

Motivation and Problem Statement

- Too much data.
- Visualization “meat grinders” not especially responsive to needs of scientific research community.
- What scientific users want:
  - Scientific Insight
  - Quantitative results
  - Feature detection, tracking, characterization
  - (lots of bullets here omitted)
- See:
  - http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf
  - http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf

Today’s Main Message

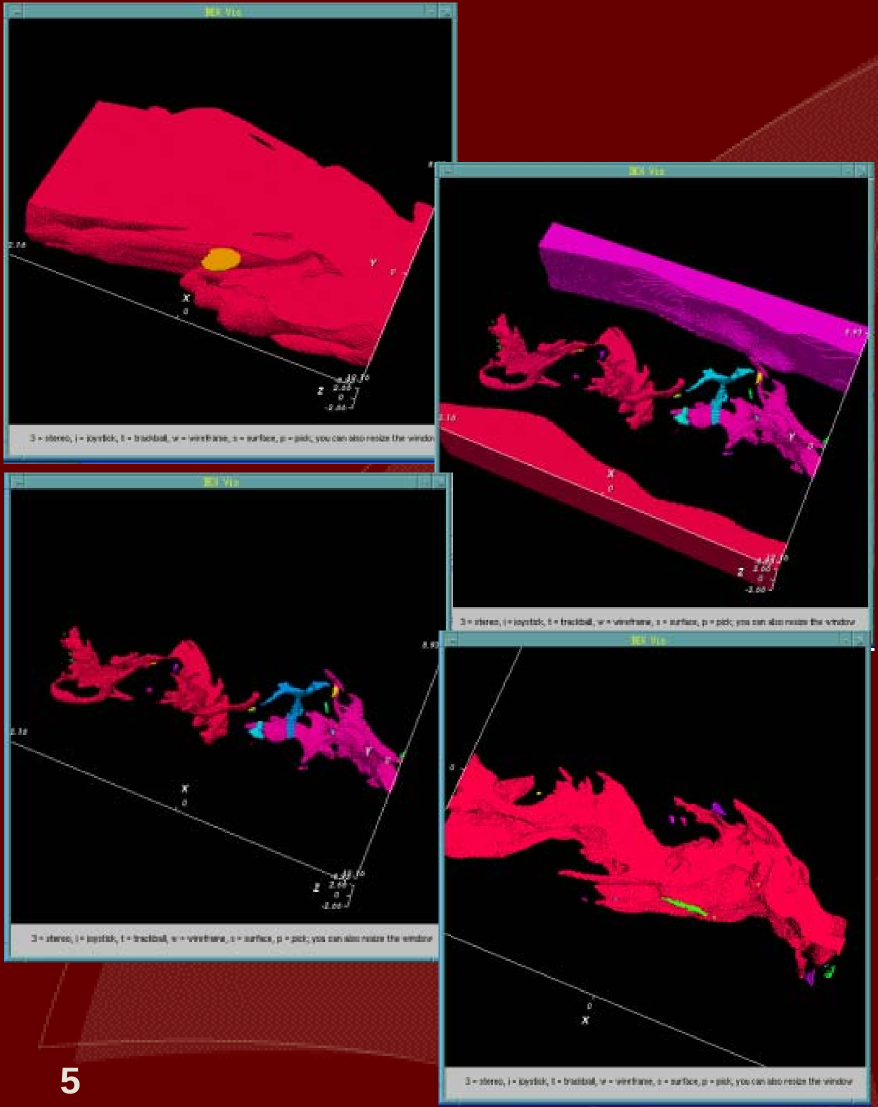
- Visualization stands to benefit in a huge way by leveraging technology from the field of scientific data management.

- An introduction to compressed bitmap indexing using reference points familiar to the visualization community.

- Compressed bitmap indexing:
  - Has low storage overhead.
  - Has low computational complexity (theoretically optimal).
  - Accommodates *n*-dimensional queries.

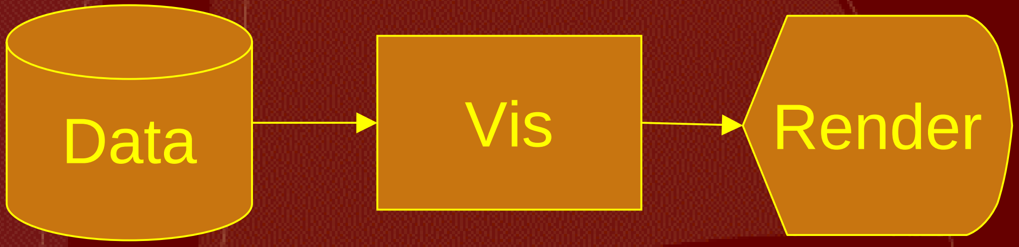
- Topics for another day:
  - Assisted/guided query posing.
  - Effective visualization of *n*-dimensional data.

Query-Driven Visualization: Visual Example

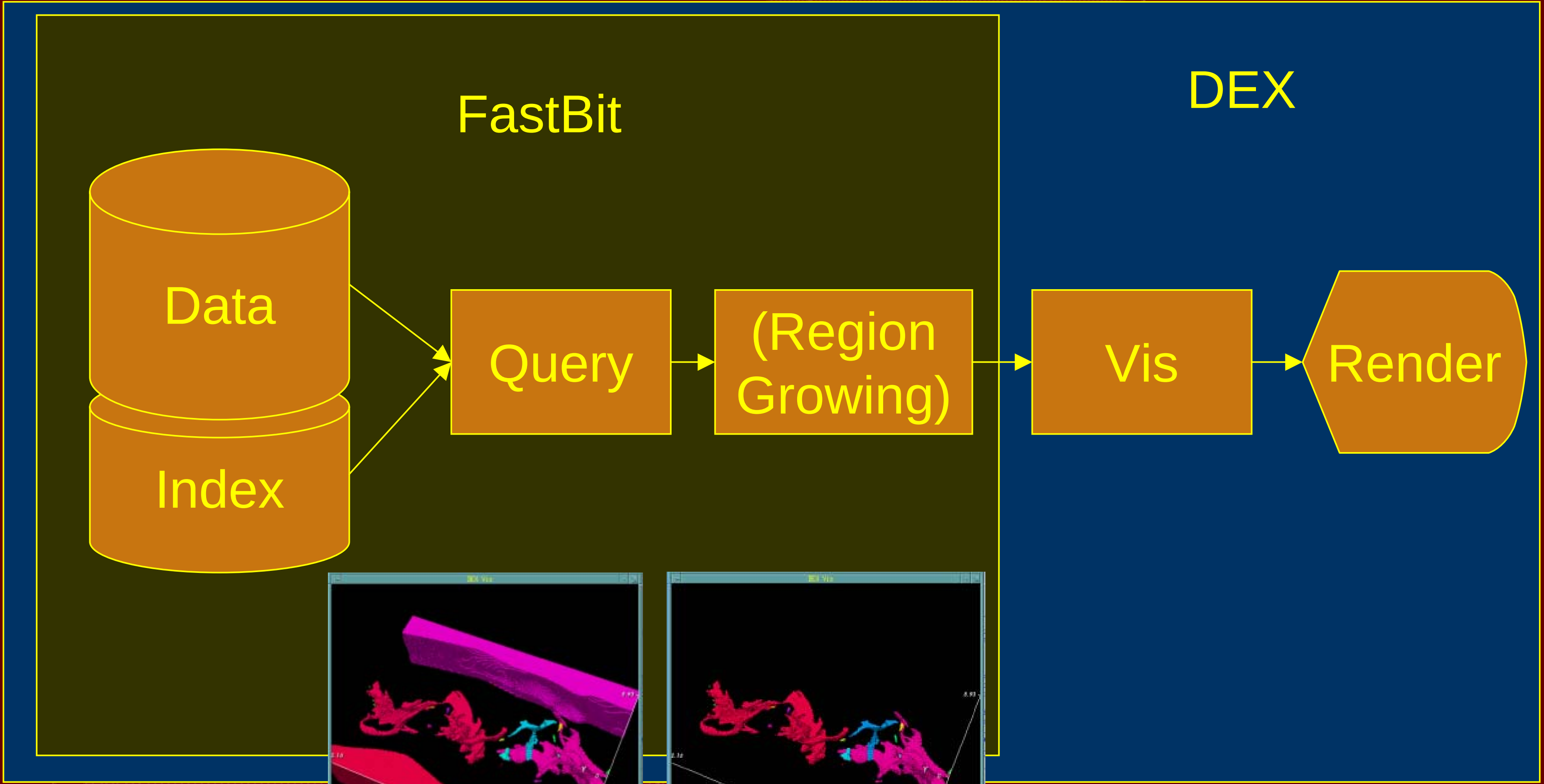


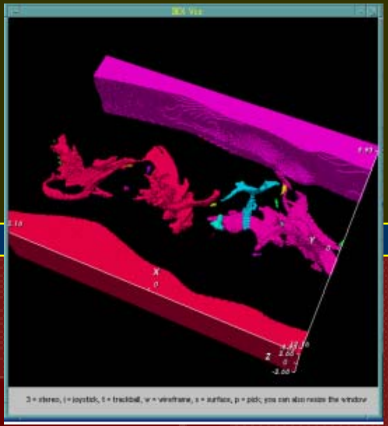
- $CH_4 > 0.3$
- $\text{Temp} < T_1$
- $CH_4 > 0.3$ AND $\text{temp} < T_1$
- $CH_4 > 0.3$ AND $\text{temp} < T_2$
  - $T_1 < T_2$

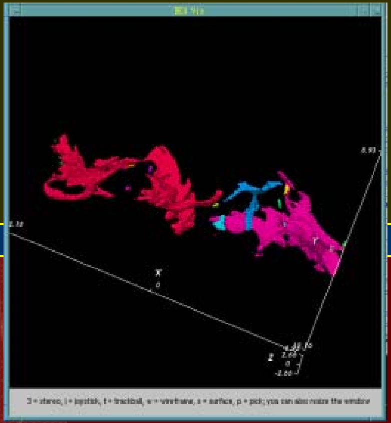
Architecture Overview: Generic Vis Pipeline



Architecture Overview: Query-Driven Pipeline




3 = stereo, j = joystick, t = trackball, w = wireframe, s = surface, p = pick; you can also resize the window


3 = stereo, j = joystick, t = trackball, w = wireframe, s = surface, p = pick; you can also resize the window

What is Query-Driven Visualization?

- Focus visualization processing on subsets of data deemed to be "interesting."
  - "Interesting" is something the user needs to define.

- Challenges
  - How to define "interesting."
    - Formulation of definition (domain-specific).
    - Expression of definition (semantic).
  - Find interesting data quickly (data management).
  - Effective visual presentation of "interesting data" (visualization).
  - Architectures/deployment that complements existing visualization algorithms and applications (computer science).

Value of Multi-dimensional Queries

- New opportunities for scientific insight: *N*-dimensional queries are the basis for complex analysis and hypothesis testing.
  - What are the characteristics of a flame front?
  - How are two (or *n*) Supernovae explosions similar/different?
  - Will this vaccine work against the Bird Flu?
  - Temporal-based queries and analysis.

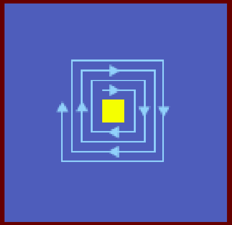
- Reducing processing and interpretation load.
  - 100TB datasets being queued up now.
  - Increased spatial resolution.
  - Lots more variables per cell.
  - Can’t expect a user to visually process 100TB of data.

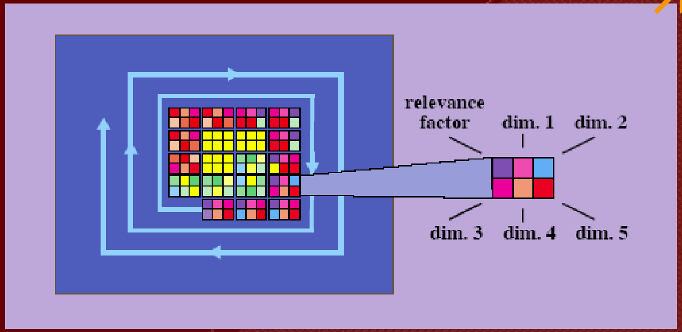
Related Work

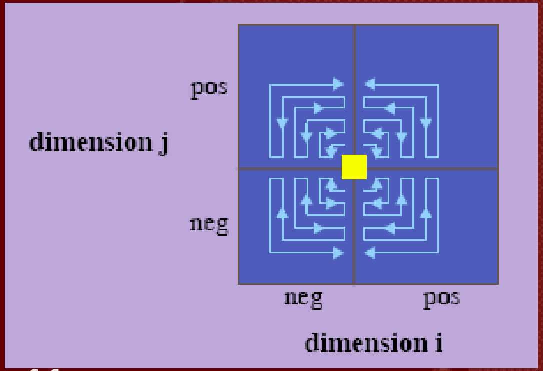
- Query-Driven Visualization
  - VisDB – Keim & Kriegel, 1994.
  - Demand Driven Visualization. Moran & Henze, 1999.
  - Scout – McCormick et. al., 2004.

- Finding Data Quickly
  - Traditional: decades of data management research.
  - Visualization community: isocontouring algorithms:
    - Marching cubes
    - Octrees – Wilhelms & Gelder, 1992.
    - Span-space methods:
      - – NOISE – Livnat, et. al., 1996.
      - – ISSUE – Shen, et. al., 1996.
      - – Interval Tree – Cignoni et. al., 1996.

VisDB – Keim and Kriegel, 1994

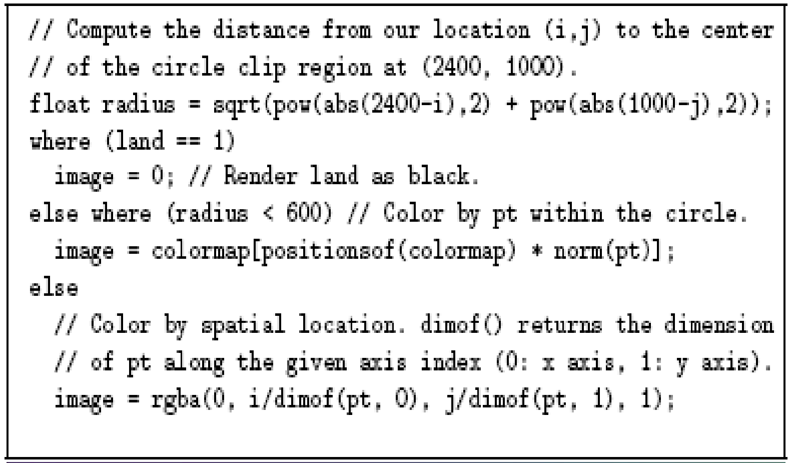




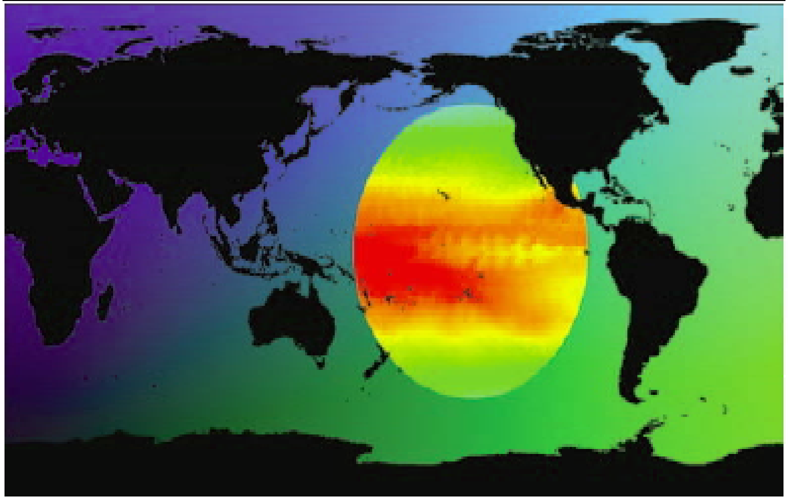


- Motivation: assist in specification of query formulation.
- Approach: rank-ordered query results.
- How:
  - For each data point [i], compute a “relevance factor” indicating how closely data point [i] matches the query (distance).
  - Sort all relevance factors, display in sorted relevance order or by colorizing relevance ranking.
- For n data values:
  - O(*n*) complexity for queries.
  - O(*n* log *n*) for sort.

Scout – McCormick et. al., 2004



```
// Compute the distance from our location (i,j) to the center
// of the circle clip region at (2400, 1000).
float radius = sqrt(pow(abs(2400-i),2) + pow(abs(1000-j),2));
where (land == 1)
  image = 0; // Render land as black.
else where (radius < 600) // Color by pt within the circle.
  image = colormap[positionsof(colormap) * norm(pt)];
else
  // Color by spatial location. dimof() returns the dimension
  // of pt along the given axis index (0: x axis, 1: y axis).
  image = rgba(0, i/dimof(pt, 0), j/dimof(pt, 1), 1);
```



- Motivation: interactive, expression-based queries.
- How: data-parallel language that executes on the GPU.
- For *n* data points, O(*n*) complexity.
- *N* will be small, though: limited GPU memory.
- Other: floating point resolution on the GPU.

Query-Driven Visualization: Summary

- Demand-Driven Visualization:
  - Visualization routines request only the data they need.
  - Works well in some circumstances: streamlines, etc.
- VisDB:
  - O(*n*) processing time for each query.
  - Data presented in relevance order, reduced in part by quartile culling.
  - Helpful for guiding queries.
- Scout:
  - O(*n*) processing time for each query.
  - High performance (GPU-based) subsetting, expressive data-parallel language.
  - Limited memory, floating-point resolution.
  - Output is imagery rather than data suitable for external use.

Finding Data Quickly

- Isosurface algorithms:
  - Nice summary in: Sutton et. al., A Case Study of Isosurface Extraction Algorithm Performance *2nd Joint Eurographics-IEEE TCCG Symposium on Visualization*, May. 2000
  - For *n* data values and *k* cells intersecting the surface:
    - Marching Cubes: O(*n*)
    - Octtree methods: O(*k* + *k* log (*n*/*k*))
      - – Acceleration: pruning; sensitive to noisy data.
    - Span-space methods:
      - – NOISE: O(sqrt(*n*) + *k)*
      - – ISSUE: O(log *(n/L*) + sqrt*(n)/L + k)*
        - » *L* is a tunable parameter
      - – Interval Tree: O(log *n* + *k*)

Finding Data Quickly: Tree-Based Methods

These approaches work well for isocontouring, but users want more than isosurfaces:

- These queries are for a single variable.
  - Want multi-valued queries. Current simulations produce 10s-100s of variables per cell.
- These queries only find cells that contain the isovalue.
  - Probably want interior cells for quantitative analysis.
- What about combinatorial tree-based methods?
  - <u>Curse of dimensionality</u>: adding more dimensions results in an exponential growth in storage and processing complexity.
  - Just say no to “n”.

Finding Data Quickly: Why Bitmap Indices

- In the data management community, the bitmap indices have supplanted trees for “heavy lifting” queries.
- Bitmap indices do not suffer from curse of dimensionality.
- Bitmap indices used in all major commercial database systems.

- Caveat: Bitmap indexing is not the panacea for everything:
  - Spatial vs. Data-value partitioning: visibility culling.

What is a Bitmap Index?

Data values	b_0	b_1	b_2	b_3	b_4	b_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
5	0	0	0	0	0	1
3	0	0	0	1	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
0	1	0	0	0	0	0
4	0	0	0	0	1	0
1	0	1	0	0	0	0

- Compact: one bit per distinct value per object.
- Easy and fast to build: O(*n*) vs. O(*n* log *n*) for trees.
- Efficient to query: use bitwise logical operations.
  - $(0.0 < H_2O < 0.1)$ AND $(1000 < \text{temp} < 2000)$
- Efficient for multidimensional queries.
  - No “curse of dimensionality”
- What about floating-point data?
  - Binning strategies.

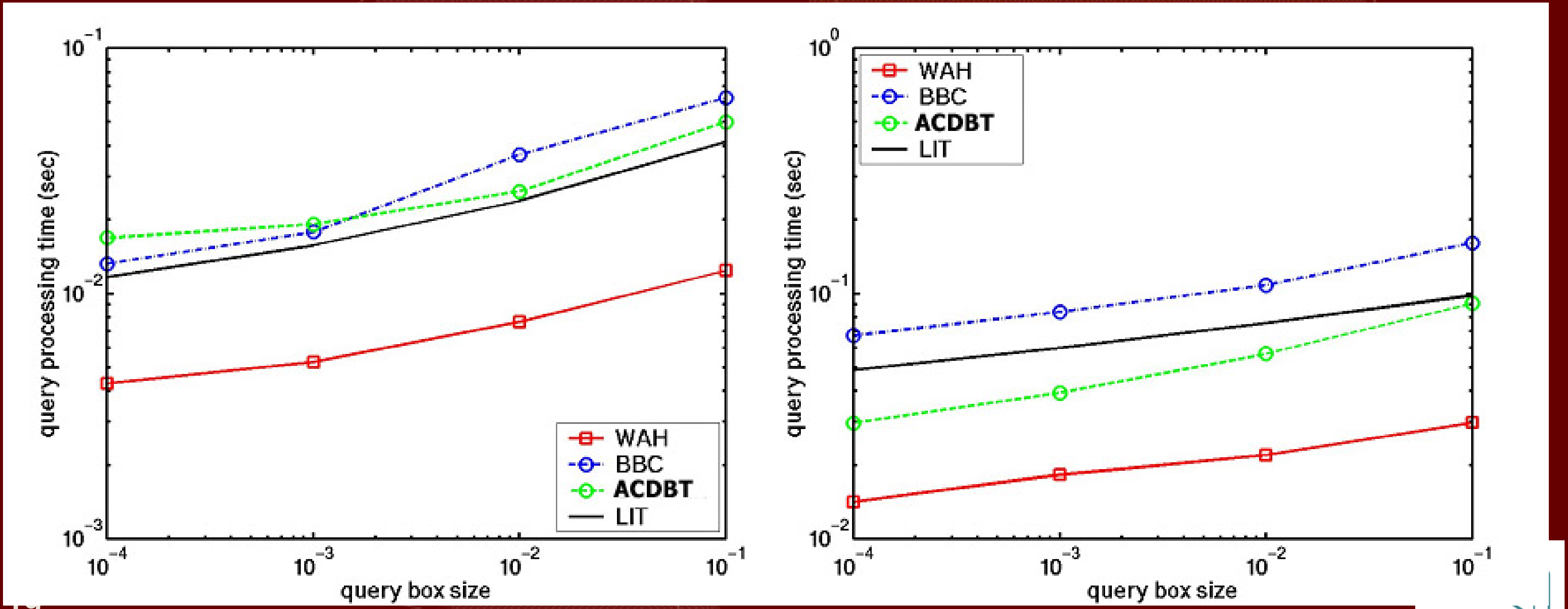
Bitmap Index Query Complexity and Space Requirements

- How Fast are Queries Answered?
  - Let N denote the number of objects and H denote the number of hits of a condition.
  - Using uncompressed bitmap indices, search time is O(N)
  - With a good compression scheme, the search time is O(H) – the theoretical optimum.

- How Big are the Indices?
  - In the worst case (completely random data), the bitmap index requires about 2x in data size (typically 0.3x).
  - In contrast, 4x space requirement not uncommon for tree-based methods.
  - Curse of dimensionality: for N points in D dimensions:
    - Bitmap index size: O(N*D)
    - Tree-based method: O(N**D)

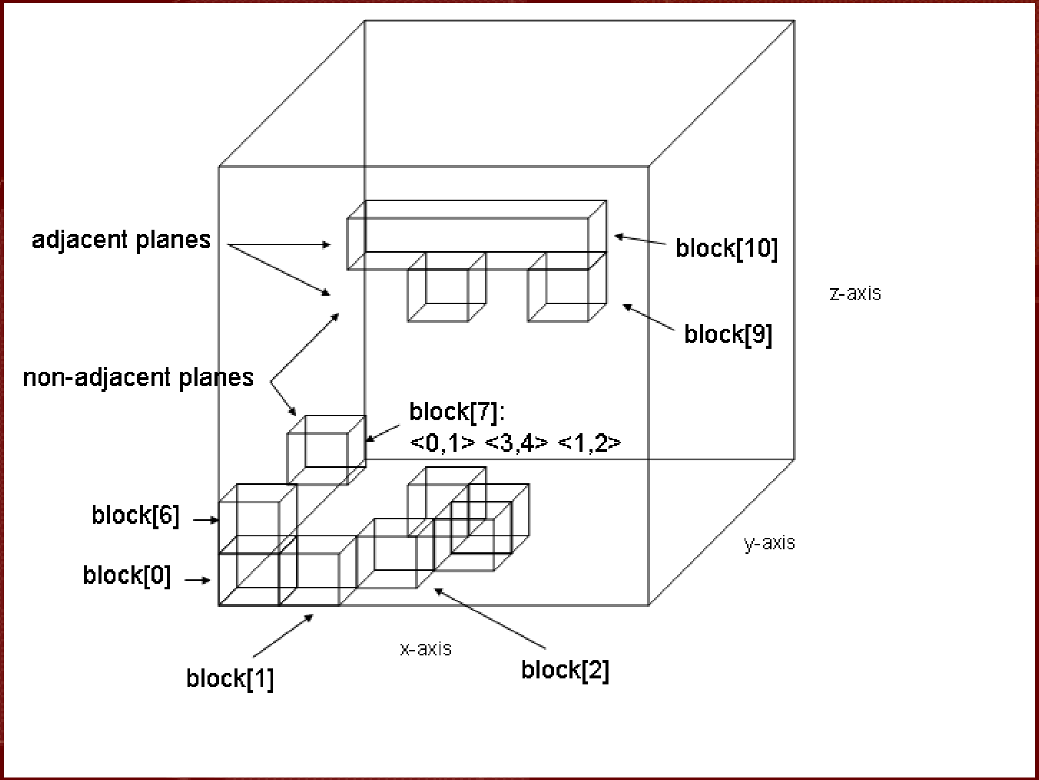
Compressed Bitmap Index Query Performance

- Different bitmap compression technologies have different performance characteristics.
- FastBit compression performance better than commercial systems.



Consolidating Query Results: Region Growing

- Find and label cells that share an edge, face or vertex.
- Not strictly necessary for “meat grinder” visualization.
- Imperative for meaningful analysis operations.



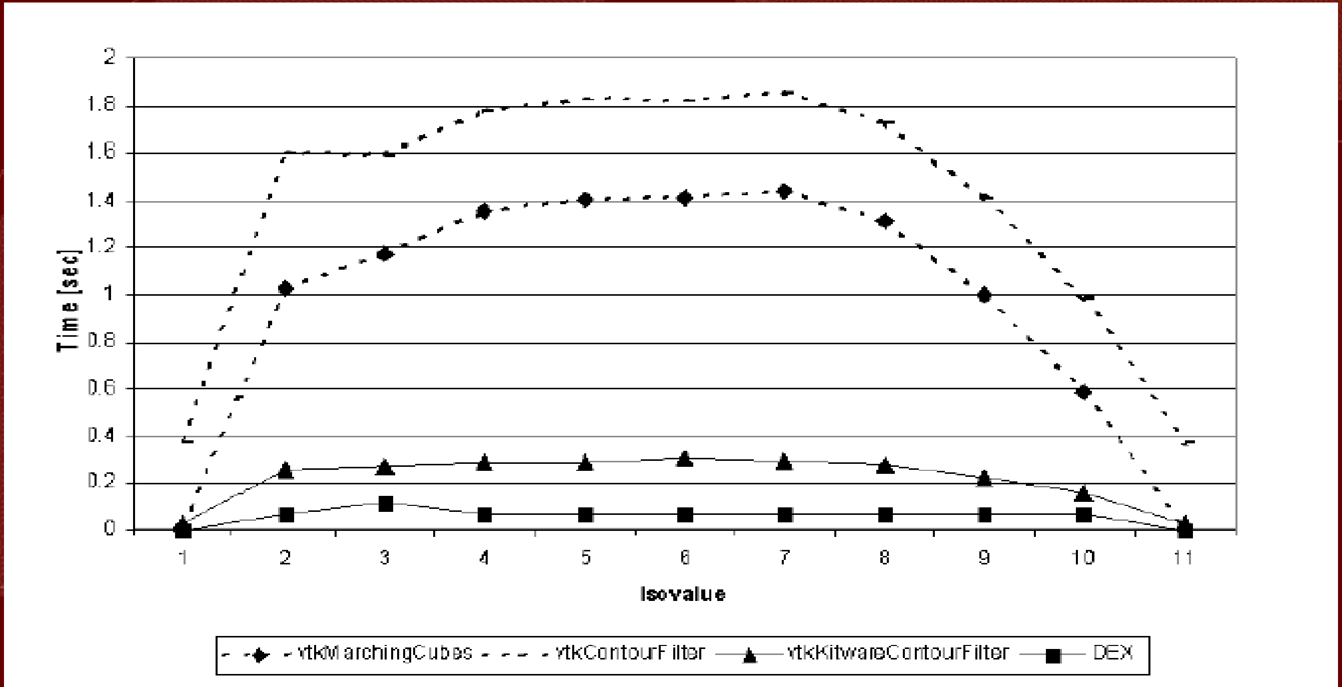
Performance Analysis Experiment

- The performance experiment:
  - Compare speed of answering queries: FastBit vs. an "industry standard isosurface implementation."
  - Note: these are queries of a single condition.

- Experimental methodology.
  - Isosurface: find cells, construct geometry.
  - DEX: find cells, construct geometry.
  - For each implementation:
    - Load dataset, disregard time required for one-time initialization.
    - For several different isovalues, measure time required to find cells and generate geometry.

Experimental Methodology, ctd.

- Which Isosurface algorithm?
  - vtkKitwareContourFilter
- Why That One?
  - It was the fastest of the VTK isocontouring algorithms (v4.4 CVS).
  - We wanted our experiments to be reproducible.

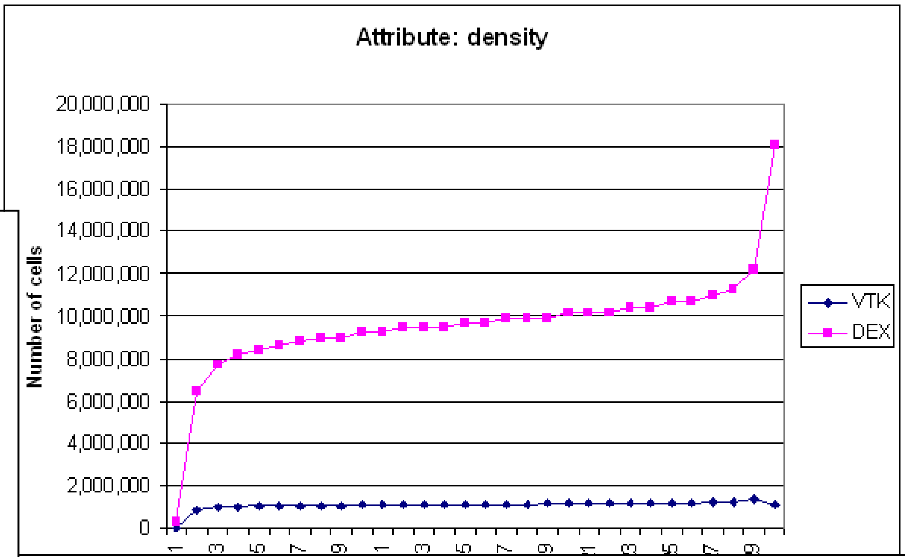


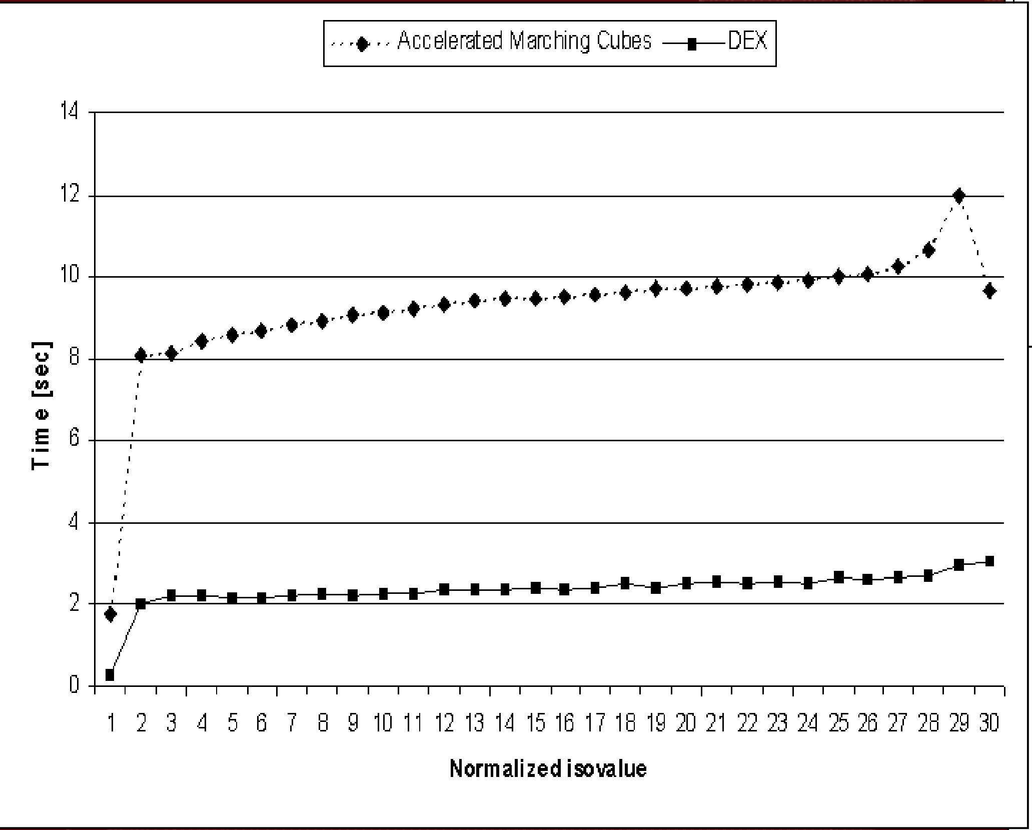
Experimental Data and Equipment

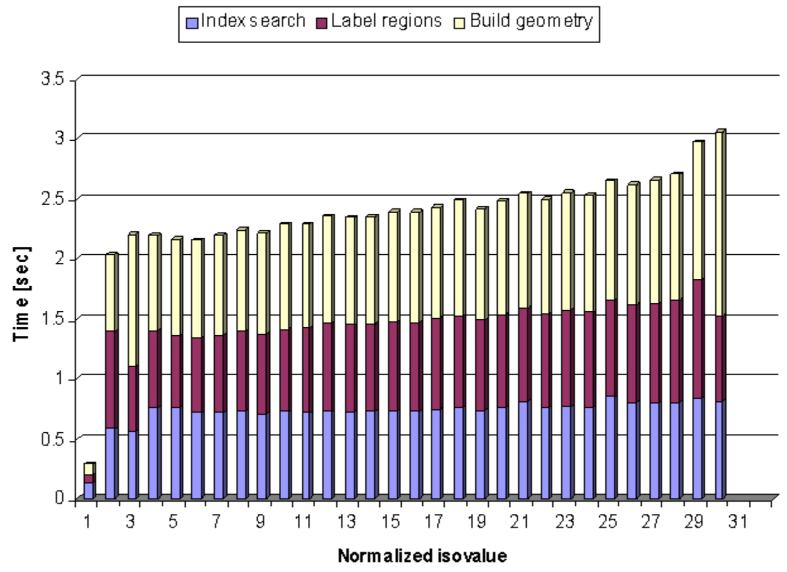
- Data
  - Results of combustion simulation.
  - Grid size: 383x383x383x38 variables.
  - “Small grid” resolution chosen to avoid impact of swapping.

- Machine
  - 2.8Ghz P4, 2GB RAM
  - 2-disk SCSI RAID, 60MB/s I/O bandwidth.

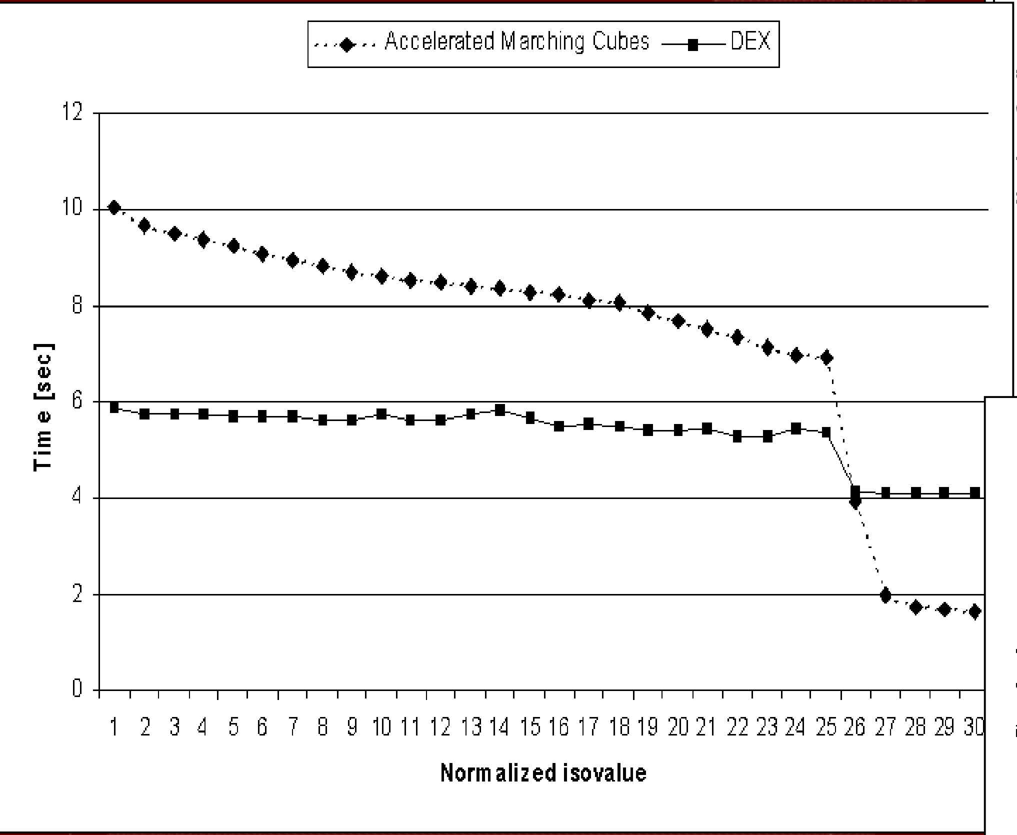
Query Performance (Density)

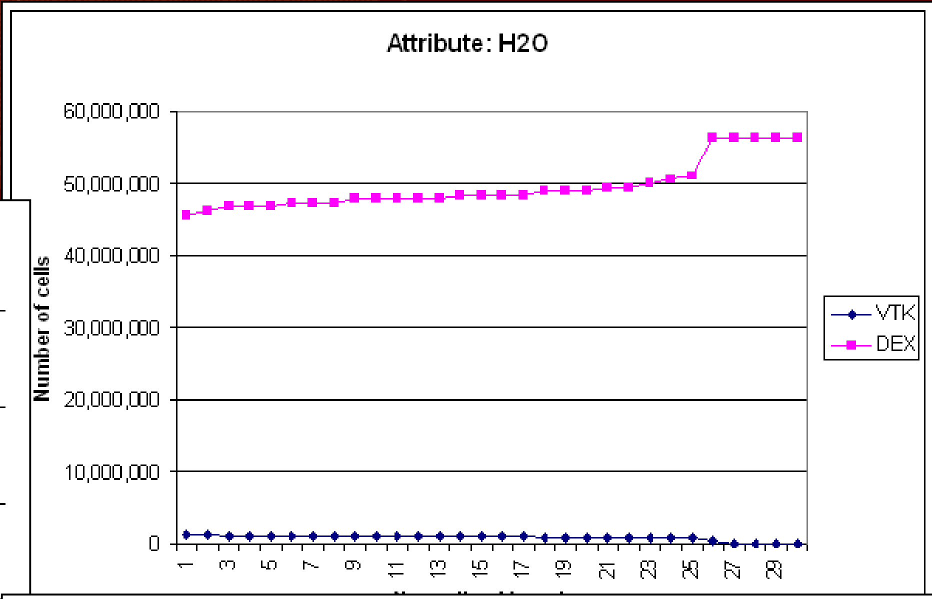


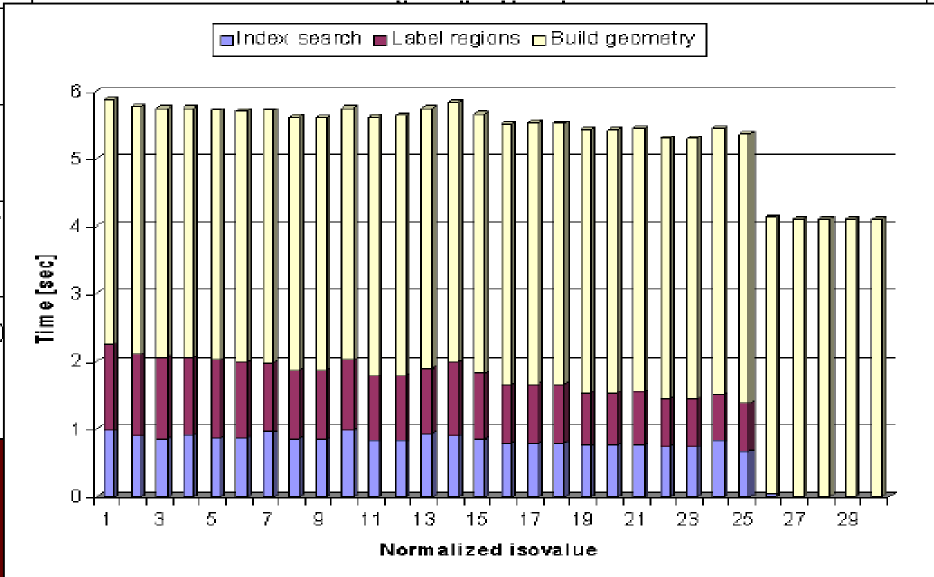




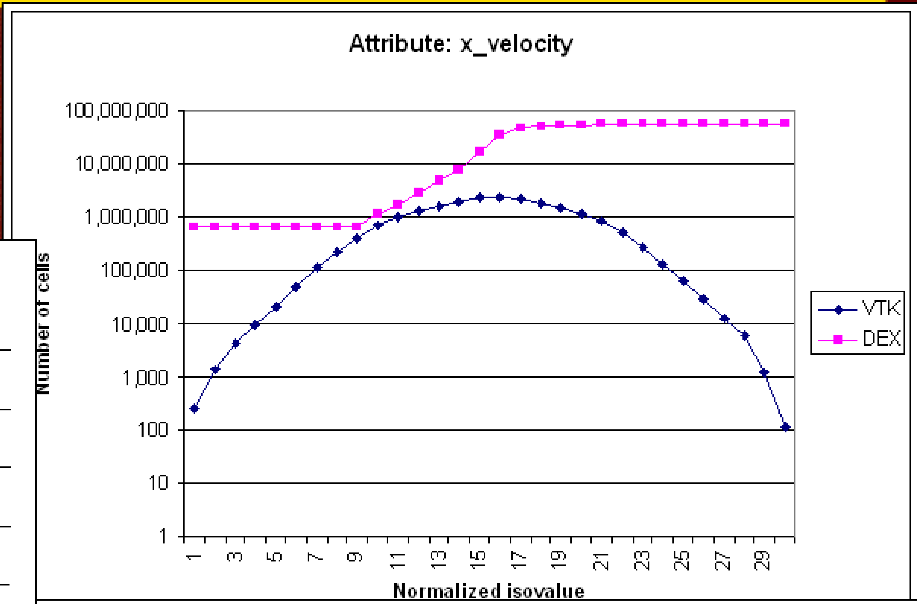
Query Performance (H_2O)

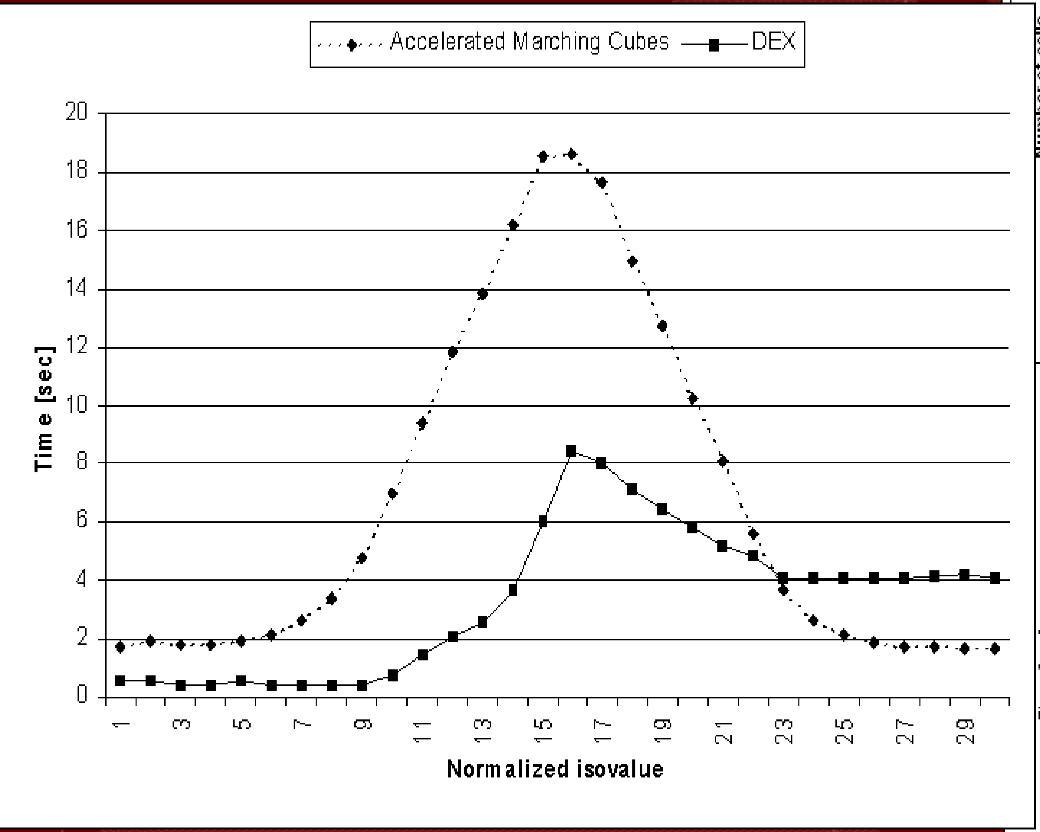


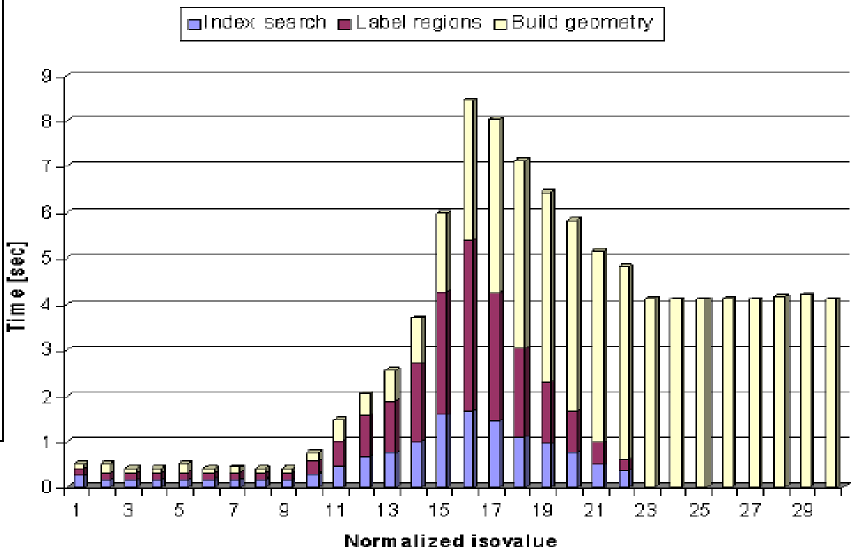




Query Performance (X-Velocity)







Discussion

- What do these timing results mean?
  - In a one-sided matchup (DEX doing a lot more work), our performance results are markedly better for a given task than an industrial-standard isocontouring implementation.
- These are single-valued queries.
  - DEX capable of *n-dimensional* queries.
  - Tree-based indexing methods not capable of *n-dimensional* queries.
- Why compare against isosurfacing?
  - Familiar to the visualization community.

Conclusion and Summary

- The Visualization Community stands to reap huge benefits by leveraging state-of-the-art technology from the scientific data management community.
  - Our study shows markedly favorable performance in single-valued queries.
- Query-driven visualization is all about supporting hypothesis testing and fostering scientific insight.
  - Quickly answering multidimensional queries is a key technology.
- DEX architecture amenable to use in a general way for visualization, analysis, …
- This approach offers new traction on the task of helping meet the needs of the scientific research community.
  - Focus vis processing and human interpretation on relevant data.
  - Fast: multidimensional queries suitable for use with multi TB data.

Future Directions

- Include in mainstream visualization tools.
  - Existing use in ROOT package from CERN.
  - AVS/Express module under development.
- Parallel implementation.
  - SC05 HPC Analytics Challenge – Network Connection Data Analysis.
    - ~2200 seconds to answer 5-D query with “industry standard”, 309 seconds with FastBit/DEX.
    - Parallel implementation: 12x parallel returns answer in 23 seconds.
- Better visualization of query results.
- Coupled analysis and vis of query results.
- Help users pose queries, iterative queries over derived variables.
- Constraints relaxation based upon proximity (space, data values, …).

The End

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Questions

- Answers (thank you J. Stasko for inspiration):
  - Yes.
  - No.
  - Maybe.
  - RTFM.
  - Hmmm, good question. Let me think about that a minute…
  - Decimation, sampling, compression, topology, rendering, …
  - That is a crazy question. Please sit down.
  - Updoc.

Index Sizes for our Performance Study

- Original data: 383^3 grid of 4-byte floats: ~215MB

Variable	Index Size (MB)	Size Factor	Time (sec)
Pressure	77.59	0.36	7.47
Density	128.70	0.60	8.56
Temperature	124.93	0.58	8.76
Velocityx	247.49	1.15	13.30
H2O	263.64	1.23	13.04
CH4	314.88	1.46	13.49

Experimental Methodology Notes

- Cell count load.
  - Isosurface: finds and processes cells that intersect the surface.
  - DEX: finds and processes cells that intersect the surfaces AS WELL AS ALL INTERIOR CELLS.
  - DEX is finding and processing 0.5-5 orders of magnitude more cells.
  - Query results are packaged differently between ISO and DEX.

- Per-Cell work.
  - Isosurface does about 1.5x more memory accesses per cell than DEX (caveat).
  - Isosurface does about 36 FLOPS/cell: ~50Mcells/sec on 2.0Ghz P4.
    - Memory access better predictor of performance (Snavely, SC05).

- Net result:
  - DEX is doing a lot more work in the performance study.
  - DEX performance is superior in nearly all test cases despite these handicaps.

Experimental Methodology Notes

- Is vktKitwareContourFilter the fastest possible implementation?
  - Possibly not, but it is what we have access to: you can reproduce our results.
  - It is the fastest of all VTK algorithms we tested.
  - It shows speedup characteristics similar to ISSUE, NOISE, etc. as compared to MC in Sutton et. al., 2000.

How Much Work Per Cell?

- Isocontouring:
  - Read 192 bytes: 3 xyz floats and 3 xyz data gradients for 8 cell corners.
  - Flops: about 96 per cell to compute triangle vertices: (assume 18 Flops per edge, and 6 edges for 2.5 triangles average per cell).
    - T = (Vone – isoLevel)/(Vone – Vzero)
    - Vnew = Vzero + t*(Vone – Vzero)
  - Write 60 bytes: 2.5 triangles * 3 coords * 4 bytes/coord for each of normals and vertices.
- DEX
  - Read load varies: output of search is (I,J,K) and (iSize, jSize, kSize) per group of cells. Worst case: 24 bytes/cell when run size is 1.
  - Flops: none.
  - Write approx 136 bytes: cell type (1 int), cell data (1 float), cell vertex indices (8 ints), 8 xyz float vertices.

How Much Work per Cell, ctd.

- Isosurface does about 1.5x more memory I/O per cell than DEX (Iso A is standard MC, Iso B reuses edge data)
- Memory access a better prediction of overall performance than just FLOPs (Snavely, SC05 paper).
- Modern CPUs perform multiple FLOPS per clock.

	DEX	Iso A, B	Comment
I/O	Worst: 160 Best: 136	252, 156	Bytes/cell
Flops	0	72, 36	Iso A: ~28M cells/sec ~2 sec to process all 53M cells on 2.0Ghz CPU

Experimental Methodology

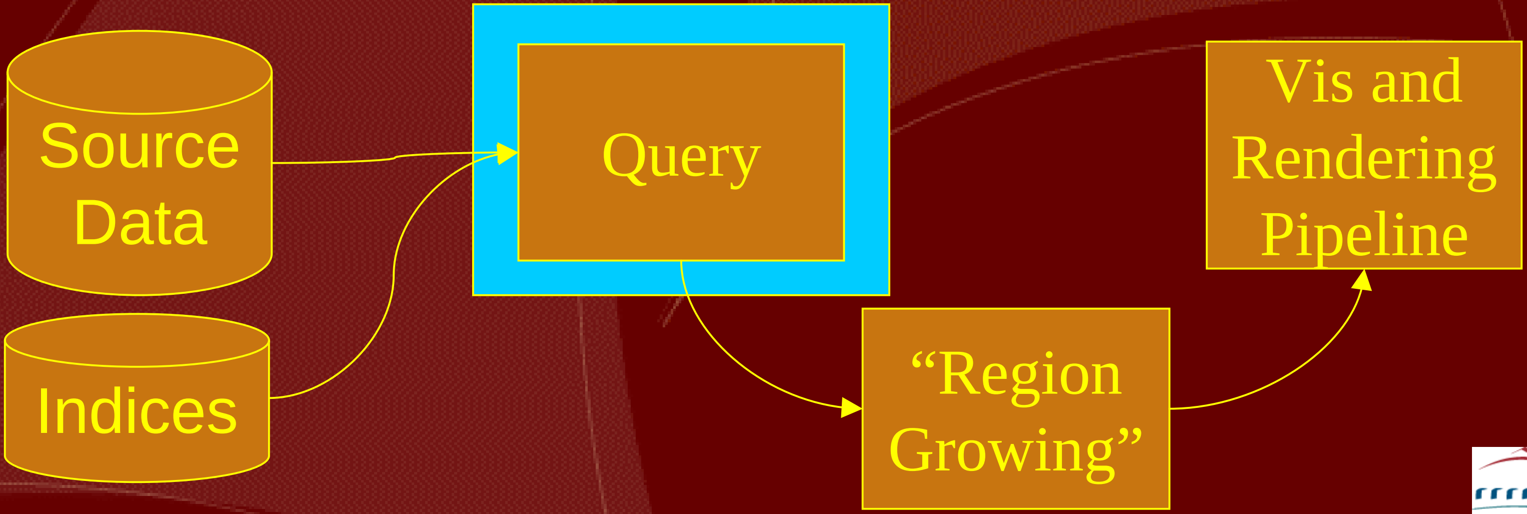
- Is including geometry construction time valid?
  - Yes. See Sutton et. al., A Case Study of Isosurface Extraction Algorithm Performance *2nd Joint Eurographics-IEEE TCCG Symposium on Visualization*, May 2000.

Experimental Methodology, ctd.

- How does geometry construction phase differ between isocontouring and DEX?
  - Isosurfacing:
    - Each cell containing the surface generates between 1-*n* triangles, where *n* varies between 4-10 depending upon the implementation.
    - Experimental results show an average of about 2.5 triangles/cell.
    - Some math required to produce triangles. FLOPS not a significant factor.
  - DEX:
    - Each cell satisfying search criteria is visually represented as a cube composed 12 triangles.
    - No math required to produce triangles.
    - In our experiments, DEX is returning the *interior* cells as well.
    - We include time for region growing in our overall time.
- Net result:
  - DEX is doing a lot more work in the performance study.
  - DEX performance is superior in all test cases despite these handicaps.

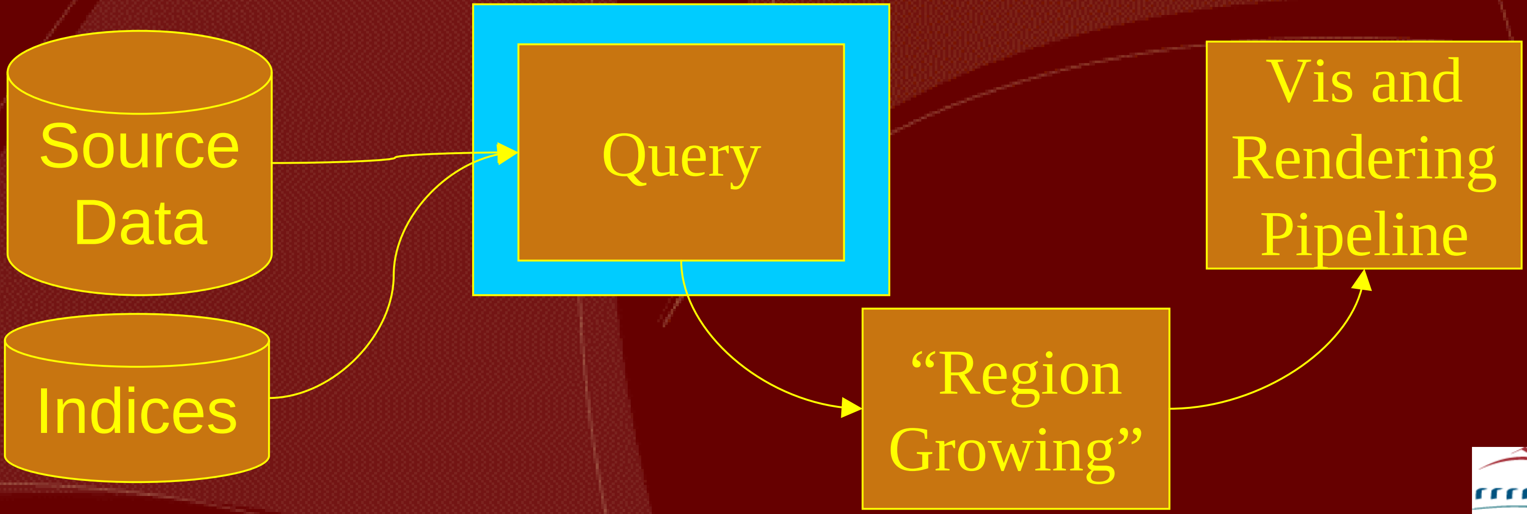
Overview of Our Implementation and Results

- Bitmap indices: the indexing structure and query engine.
  - See http://sdm.lbl.gov/fastbit
  - State-of-the-art from the scientific data management community.
- Preprocessing query output.
- Provide to visualization engine.
- Experimental performance results.



Overview of Our Implementation and Results

- Bitmap indices: the indexing structure and query engine.
  - See http://sdm.lbl.gov/fastbit
  - State-of-the-art from the scientific data management community.
- Preprocessing query output.
- Provide to visualization engine.
- Experimental performance results.


Source Data
Indices
Query
"Region Growing"
Vis and Rendering Pipeline